Now that you're beginning to get familiar with the controls and operating aspects of the computer, it's time to get into a real program. After all, reading a number out of memory and incrementing it is not the kind of thing that got you interested in microcomputers to begin with. So let's try a real program, something useful that we can play with and expand. While the task is ridiculously simple it will serve to illustrate some very important aspects of microcomputer programming. It also happens to be an application problem that virtually every microcomputer system is faced with.

The question is deciding whether a switch is open or closed. The fact that we are even raising the question is hard evidence of the fact that microcomputers are drastically different from conventional digital logic systems. In a conventional system the switch would simply be connected to one of the logic elements at an appropriate place in the system. The switch would then provide either a high or low voltage to that element and the question of whether the switch was open or closed would be decided accordingly. A change in the state of the switch would be instantly translated into a logic change which would cause corresponding changes to occur within the system. In the microcomputer, the switch is connected to one of the inputs to the system. When a point in the program is reached where the CPU needs to know the status of the switch the program will test it. The information as to whether the switch is open or closed will be loaded into one of the registers, such as the accumulator, where it will be tested and appropriate action taken. Thus, unlike the conventional logic system, the microcomputer only tests the switch when it needs to know the state of that switch.

Because there may be many switches and other outside devices connected to the computer, we will number them so we can direct the computer which to test. That way we can test lots of switches and keep them all straight. Connections to the outside world are made through groups

of wires called ports. Input ports are used to read information into the computer and output ports are used to send information out of the computer; generally speaking a port cannot be used to both read and send data; it will be permanently designed as one or the other and cannot be changed.

There are two different systems for numbering the connections to outside devices. The numbering system that the ia7301 uses treats the ports as parts of the memory. For this reason, the ports will be assigned addresses in much the same way that the various memory locations had addresses. This technique is referred to as Memory-Mapped I/O (Input/Output). It has the

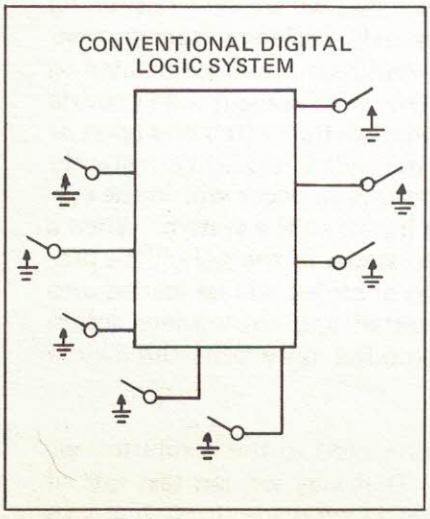


Fig. 4-1. In the conventional digital logic system, switches are connected to points scattered throughout the system. A change in the state of a switch will be immediately recognized by the logic element that monitors it, although the system as a whole may not act upon the change.

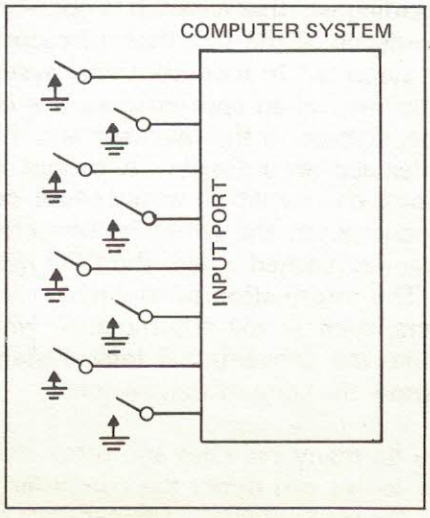


Fig. 4-2. In a computer system, all of the switches and input devices will be connected to the computer through input ports. It is up to the program being executed to direct the computer to read the switch data at the input port and how to react to that data, Between readings, the position of the switches is unknown.

advantage that a wide variety of standard instructions can be used to move information into and out of the computer. The other technique, not used by the ia7301, is a dedicated I/O system whereby only special I/O instructions can be used to move data into and out of the system. This technique does not tie up memory addresses the way the memory-mapped I/O does, but the data rate is much slower and more awkward to implement within the program. While it is true that the memory-mapped I/O technique does reduce the amount of memory available to the user, the total memory capacity is never a problem.

The ia7301 has an input port and an output port whose connections are brought out to the edge connector for your use. However, rather than slow you down by asking you to solder switches

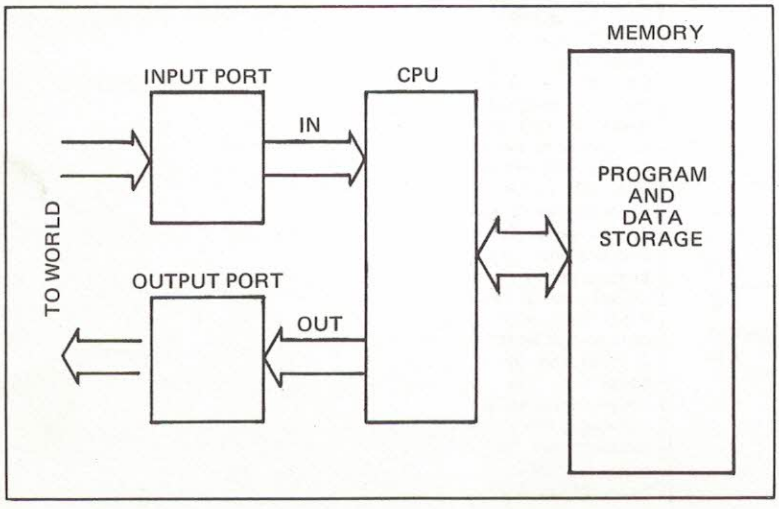


Fig. 4-3. In the dedicated I/O type of system, the CPU communicates with the I/O ports through special IN and OUT instructions. In this technique the ports are assigned port numbers, INPUT PORT 13H, etc. This leaves the entire memory free for program and data storage.

onto the edge connector for the purpose of the upcoming discussion, we have provided two switches right on the board itself. These switches can be tested by our program without having to make any special connections.

In addition to the input and output port that are tied to the edge connector, there are several other I/O ports within the ia7301 that are not brought to the edge connector. These are dedicated to specific functions within the computer like driving displays and reading keyboards. It

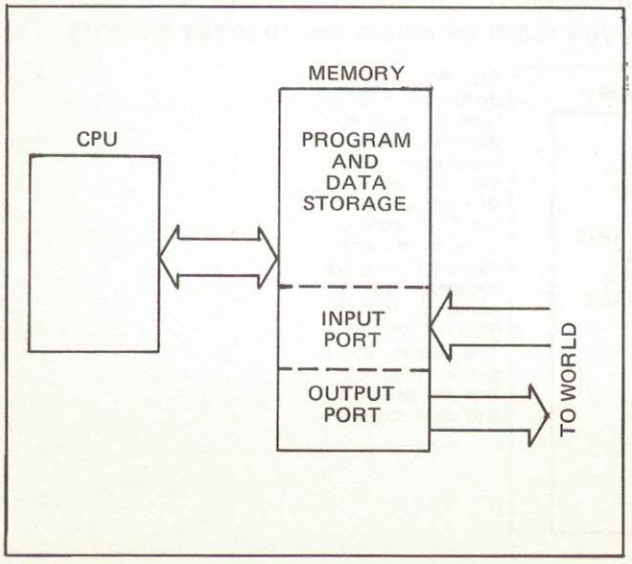


Fig. 4-4. In a memory-mapped I/O type of system like the ia7301, the I/O ports are considered to be part of memory and are assigned addresses just as though they were memory locations. This allows the programmer to use a wide variety of instructions and programming techniques to receive and send data to the world. It does use up some of the memory capacity although the reduction can be made so small as to be insignificant.

turns out that one of these input ports happened to have two unused connections, so we hooked up the switches to them so that we can experiment with them in this course. You will find the switches at the bottom left corner of the keyboard. Both switches are mounted in the same tiny plastic enclosure. The plexiglass cover of the computer has been milled out at this point to provide easy access to the switches.

Since the switches are quite small, you will probably find it convenient to use a pencil tip for operating them. Depending on the individual switch used in your particular system, you will find the switch will be either a rocker type (a miniature version of the common wall switch) or a slide type which requires that the operating handle be slid from position to position. Both types of switches fall into a category commonly referred to as DIP switches. In any event the switch will be marked in one of several ways. That marking will tell you which position is ON or CLOSED. The other position will be either OFF or OPEN. The switch is connected on the board so that the OFF position will produce a high voltage input to the computer (logic 1) and the ON position will produce a low voltage input to the computer (logic 0).

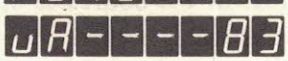
The reader unfamiliar with the concepts of digital logic and the use of high and low voltages to represent logic states will find a short description of these techniques in the supplementary material in the back of the book.

The input port that ties these switches to the computer corresponds to address CXXXH. Any memory read operation done from a memory location in the CXXXH block of addresses will cause the state of the switches to be read into the computer. The fact that the input port is assigned this large block of memory addresses prevents us from using them for actual memory locations. This is not a problem since the computer uses only addresses 000H to 03FFH and 8000H to 87FFH for memory. The rest of the addresses up the FFFFH are unused and may be dedicated to I/O ports.

We already have an instruction in our repertoire that can be used to read the state of the switches into the computer. That instruction, LDA, was used in the last chapter for reading the contents of a specified memory location into the accumulator. Now, the same techniques will cause it to read the switches for us. We will write a short program.

Enter:	See Displayed:	Write Contents:
CLR		
0 1 0 0 NXT		
3 A		
NXT 0 0		
NXT C 0		
NXT F F		
NXT DCR		

Our program is the height of simplicity. LDA C000H will cause the input port that is connected to the switches to be read into the accumulator. An FFH following this will cause the system to stop and enter the single-step mode where we can monitor the process. Before executing the program, set both of the DIP switches to the ON or CLOSED position. Then press EXC .

Enter:	See Displayed:	Write Contents:
EXC		
DCR		

If the program executed properly you should have found that the accumulator contained the data 83H. Does this mean that both switches are closed? How can we tell? Set switch 1 (the leftmost of the pair) to the OPEN or OFF position. Leave switch 2 unchanged. Repeat the program. (Don't forget to CLR so that program counter is reset to 0100H.)

Enter:	See Displayed:	Write Contents:
CLR	n- - - - - - - -	
EXC	F 0 1 0 4 -	
DCR	uA- - - - 9 3	_____

So, the accumulator has been changed by changing the position of the switches! Apparently we are reading the switch data into the accumulator although it is not obvious at this time what the data means. Change switch 2 to to OPEN or OFF position; leave switch 1 in the OPEN or OFF position. Repeat the program.

Enter:	See Displayed:	Write Contents:
CLR	n- - - - - - - -	
EXC	F 0 1 0 4 -	
DCR	uA- - - - b 3	_____

And so the accumulator has been changed again. The switches are obviously being read, but the format of the data in the accumulator is not obvious. What's more the question of how the computer can use this data still must be answered. After all, if the computer is to react to the closing of a switch it must be able to make some sort of a decision based on that data. What is that decision making process and how can we use it in our programs? Read on.